Accessing SCSI and ATA Devices in Mac OS X

Session 111

Accessing SCSI and ATA Devices in Mac OS X

Tim McLeod
Technical Lead
Mass Storage Software

Introduction

- SCSI and ATA devices in Mac OS X already provide many services that can be used by applications
- New services will be added to the next release Jaguar, including:
    - Scanner support via ImageCapture
    - Printing Support via CUPS
    - DiscRecording framework

So, what are these services, how do they differ and which one should I use?

What You Will Learn

- What SCSI and ATA devices are
- How these devices are represented in Mac OS X
- The services that are provided by these devices
- How to access these services from applications

What Is a SCSI Device

- Complies with one of the SCSI command specifications from the T10 committee
- Can be attached by any one of the supported physical interconnects such as FireWire, ATA(ATAPI), USB or SCSI Parallel

  http://www.t10.org

- In Mac OS X, the name SCSI Parallel will be used to refer to the traditional parallel SCSI bus and devices

What Is an ATA Device

- Complies with an ATA specification as defined by the T13 committee
- Such as:
    - ATA hard drives
    - PCMCIA/ATA storage devices

http://www.t13.org

Services Provided by SCSI and ATA Devices

- Storage services
- Application-specific services
- Or both

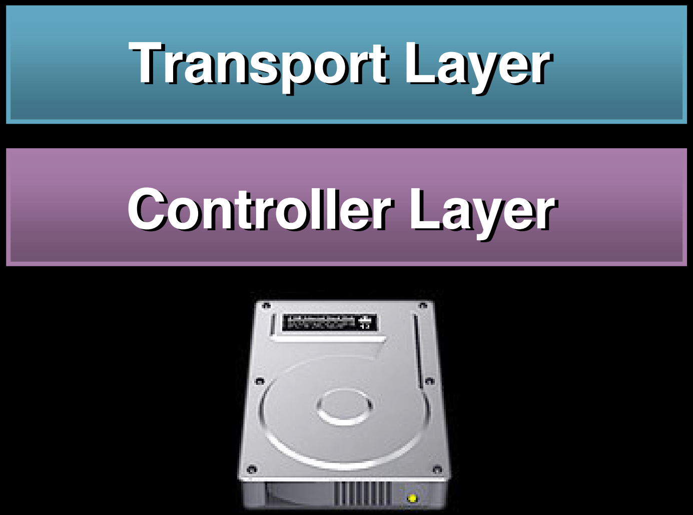
Assumptions and Limitations

- All devices are detachable from the system
- Only a single entity can control a device at any given time

General Storage Services Model

Controller Layer

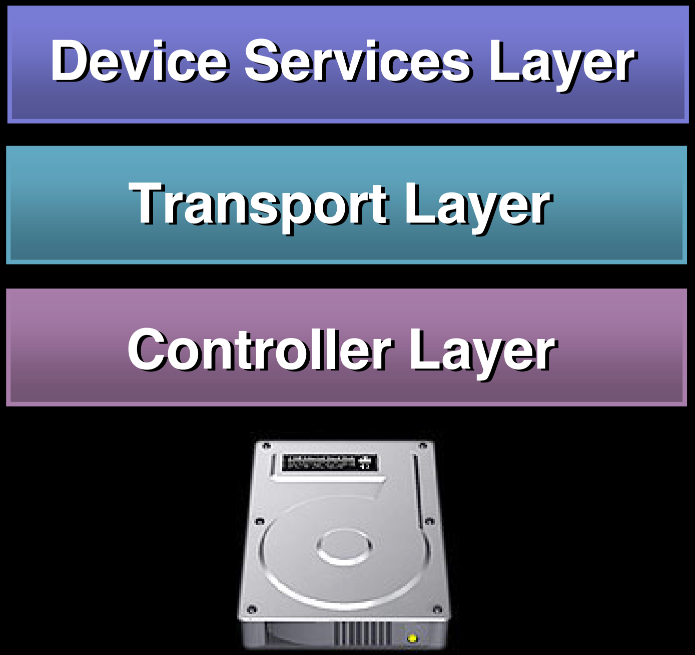
General Storage Services Model



Transport Layer

Controller Layer

General Storage Services Model

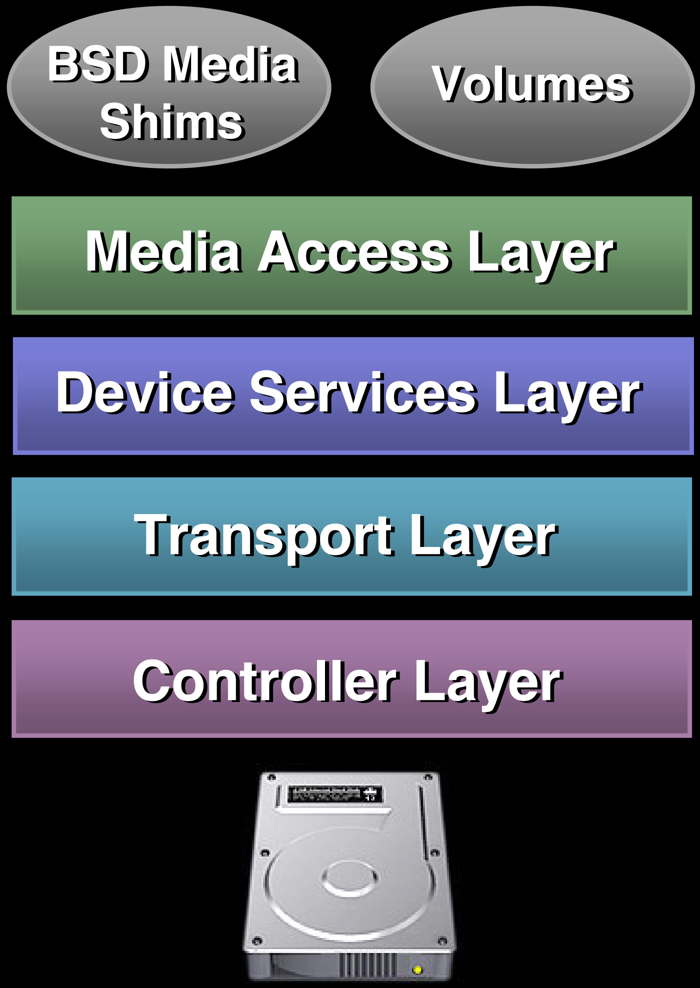


Device Services Layer

Transport Layer

Controller Layer

General Storage Services Model



BSD Media Shims

Volumes

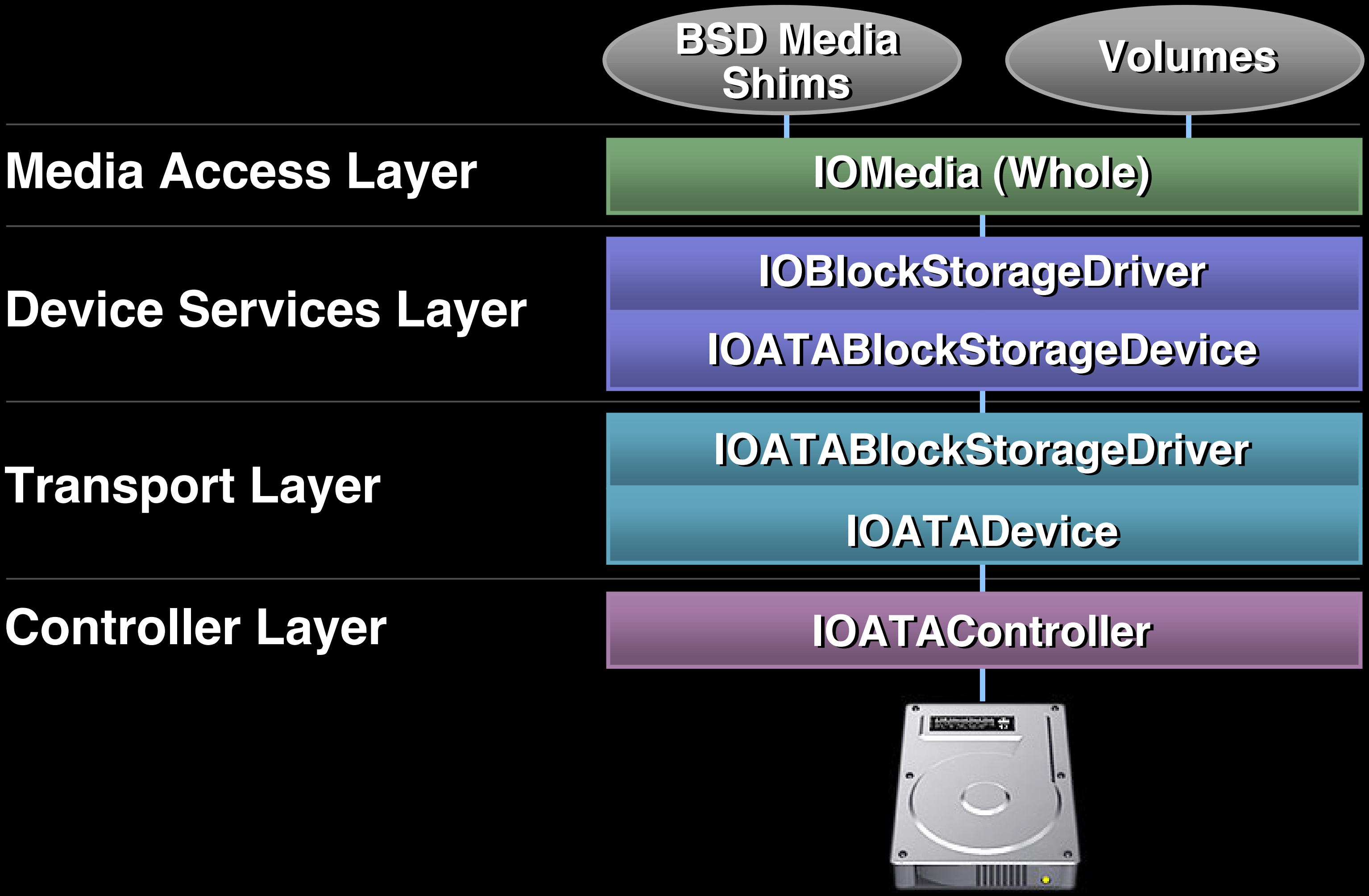
Media Access Layer

Device Services Layer

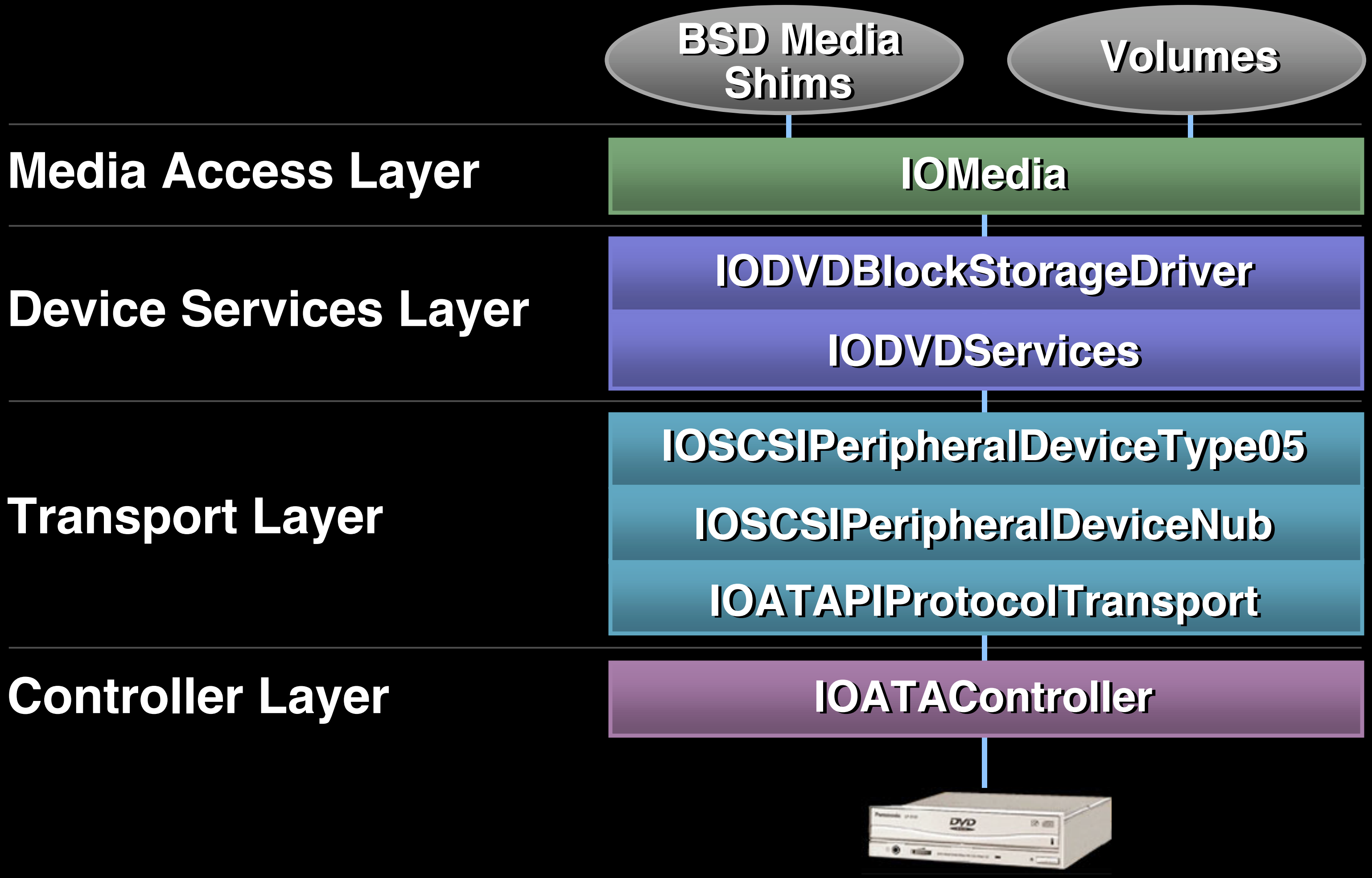
Transport Layer

Controller Layer

ATA Storage Services Model



SCSI Storage Services Model



BSD Media Shims

Volumes

Media Access Layer

IOMedia

Device Services Layer

IODVDBlockStorageDriver

IODVDServices

Transport Layer

IOSCSIPeripheralDeviceType05

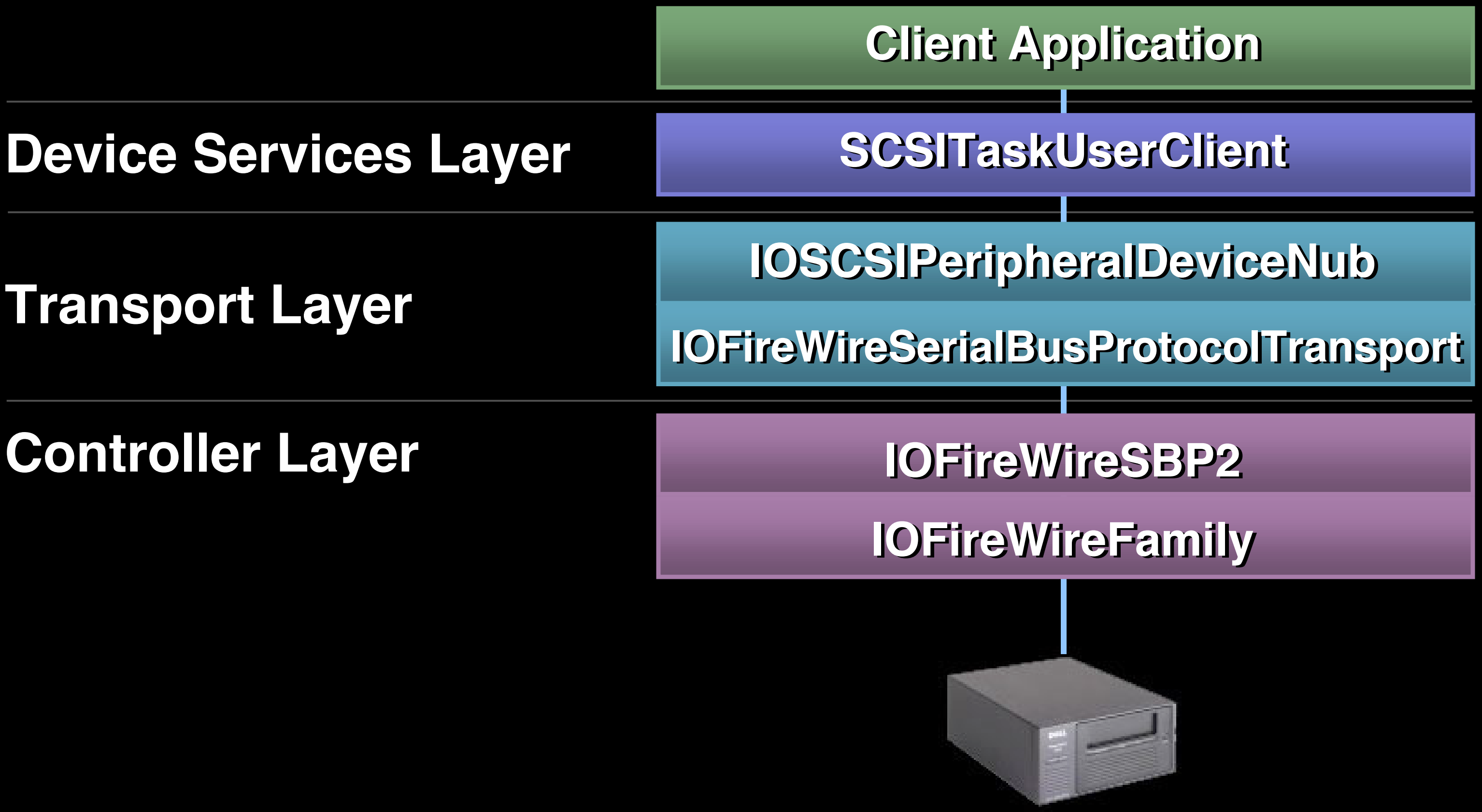
IOSCSIPeripheralDeviceNub

IOATAPIProtocolTransport

Controller Layer

IOATAController

SCSI Application-Specific Model



Client Application

Device Services Layer

SCSITaskUserClient

Transport Layer

IOSCSIPeripheralDeviceNub

IOFireWireSerialBusProtocolTransport

Controller Layer

IOFireWireSBP2

IOFireWireFamily

Methods of Access

- An application can access the provided services via:
  - Media Access Layer, BSD Media Shims, and IOMedia filters
  - User Clients, both provided and custom
  - The IORegistry
  - Migration from obsoleted methods such as IOSCSILib, IOCDBLib, and SCSIAction

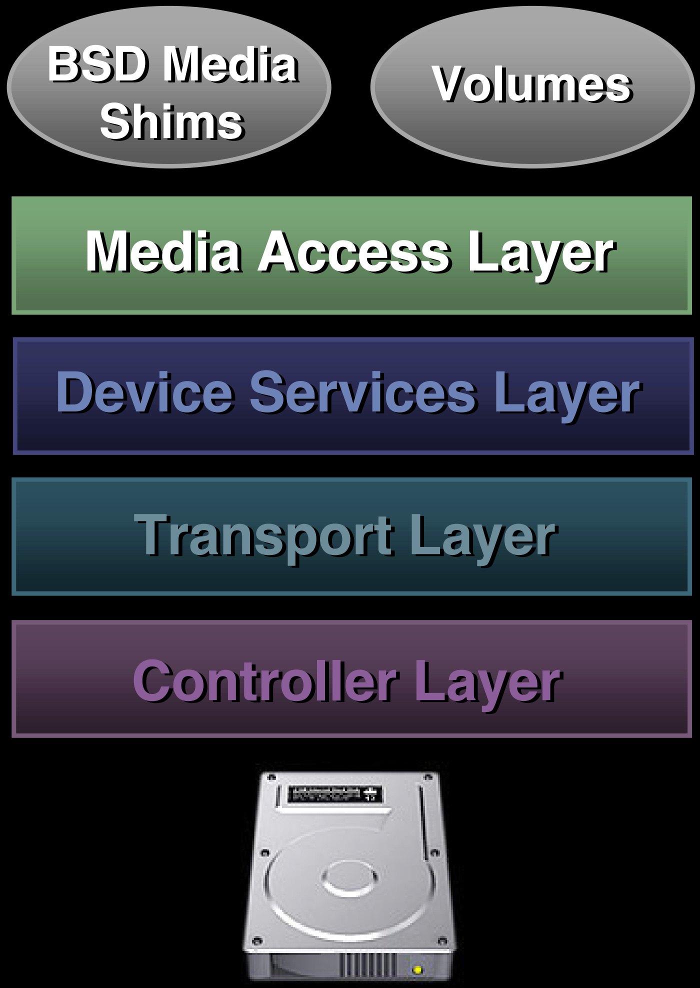
Methods of Access: Media Access Layer

Craig Marciniak
Senior Software Engineer
Mass Storage Software

What You Will Learn

- Understanding IOMedia Objects
- Using the BSD Client Interface
- Using the BSD APIs from Carbon and Cocoa
- Understanding IOMedia Filters
- Writing an IOMedia Filter

Media Access Layer



BSD Media Shims

Volumes

Media Access Layer

Device Services Layer

Transport Layer

Controller Layer

IOMedia Objects

- Object representations of contiguous logical storage
- Abstracted from hardware details
- Byte-based APIs with 64-bit parameters

The BSD Media Shim

- dev nodes are file representations of devices that are stored in /dev
- BSD device interfaces provide user-client access for IOMedia objects
- Example device nodes:
  /dev/disk0, /dev/rdisk0, etc.
- Simple five entry point API
  (open, close, read, write, and ioctl)

BSD Device Interfaces

- There are two device interfaces
  - Raw (sometimes referred to as unbuffered or character)
  - Block (sometimes referred to as cooked) which are buffered and/or processed

Raw Device Interface

- Media is accessed via /dev/rdisk* device nodes
- Access must be a multiple of the natural block size or it will be rejected with an error
- Low-level disk utilities should use this interface
- Example: a database might want to use this interface to manage their own tuned caching scheme

Block Device Interface

- Media is accessed via /dev/disk device nodes
- Access does not have to be a multiple of the natural block size since it is buffered
- File systems generally use this interface

Using the Interfaces

- Uses POSIX.1 style I/O functions
- Mode follows established UNIX semantics
- You need a BSD path to the desired node

Getting BSD Paths From Cocoa

- Cocoa's NSFileManager and NSWorkspace use mount points
- Mount points can be translated to BSD paths via getmntinfo( ) or statfs

Example

```
char                bsdPath [MAXPATHLEN] = { 0 };
int                 index, mInfoCount;
struct statfs *     mInfo;

mInfoCount = getmntinfo ( &mInfo, 0 );
for ( index = 0 ; index < mInfoCount ; index++ ) {

    if  mntonname equals mountPoint
        copy mntfromname to bsdPath
}
```

Simplified Example

```
char                bsdPath [MAXPATHLEN] = { 0 };
struct statfs *     mInfo;

if ( statfs ( mountPoint, & mInfo ) == 0 ) {

     copy mntfromname to bsdPath
}
```

Getting BSD Paths From Carbon

- Carbon uses Volume Reference Numbers
  - FSSpec, FSRef, or use FSGetVolumeInfo
  - Carbon provides a method to convert a vRefNum to a C-string which represents the BSD path

Example

```
Char                    bsdPath[MAXPATHLEN] = { 0 };
HParamBlockRec          pb;
GetVolParmsInfoBuffer volParmsInfo;

Initialize parameter blocks
pb.ioParam.ioVRefNum = vRefNum;
pb.ioParam.ioBuffer = ( Ptr ) &volParmsInfo;
pb.ioParam.ioReqCount = sizeof ( volParmsInfo );

if ( PBHGetVolParmsSync ( &pb ) == noErr ) {
    strcpy ( bsdPath, volParmsInfo.vMDeviceID );
}
```

open ( )

```
char            bsdPath [MAXPATHLEN] = { 0 };
int             fd;

fd = open ( bsdPath , O_RDONLY );

// if fd is equal to -1 see errno for error
```

- See *man* pages, *The Design and Implementation of the BSD 4.4 Operating System*, and/or Steven's *Advanced Programming in the UNIX environment*

read ( ) and write ( )

- Reads and writes from and to raw device nodes must be a multiple of natural block size

```
bytesRead = read ( fs, &buffer, readCount );
// if bytesRead = -1 check errno for details

bytesWritten = write ( fs, &buffer, writeCount );
 // if bytesRead = -1 check errno for details
```

ioctl

- Pronounced IOCTL (sometimes IO Controls)
- <IOKit/storage/IOMediaBSDClient.h>
  - <IOKit/storage/IOCDMediaBSDClient.h>
  - <IOKit/storage/IODVDMediaBSDClient.h>

- Examples:

  DKIOCEJECT, DKIOCGETBLOCKSIZE, etc.

New ioctls in Jaguar

NEW

- IOMediaBSDClient.h
  - DKIOCSYNCHRONIZECACHE *Flush write cache*
- IOCDMediaBSDClient.h
  - DKIOCCDREADTOC *Read TOC*
  - DKIOCCDREADDISCINFO *Read disc info*
  - DKIOCCDREADTRACKINFO *Read track info*
- IODVDMediaBSDClient.h
  - DKIOCDVDREADDISCINFO *Read disc info*
  - DKIOCDVDREADRZONEINFO *Read RZone info*

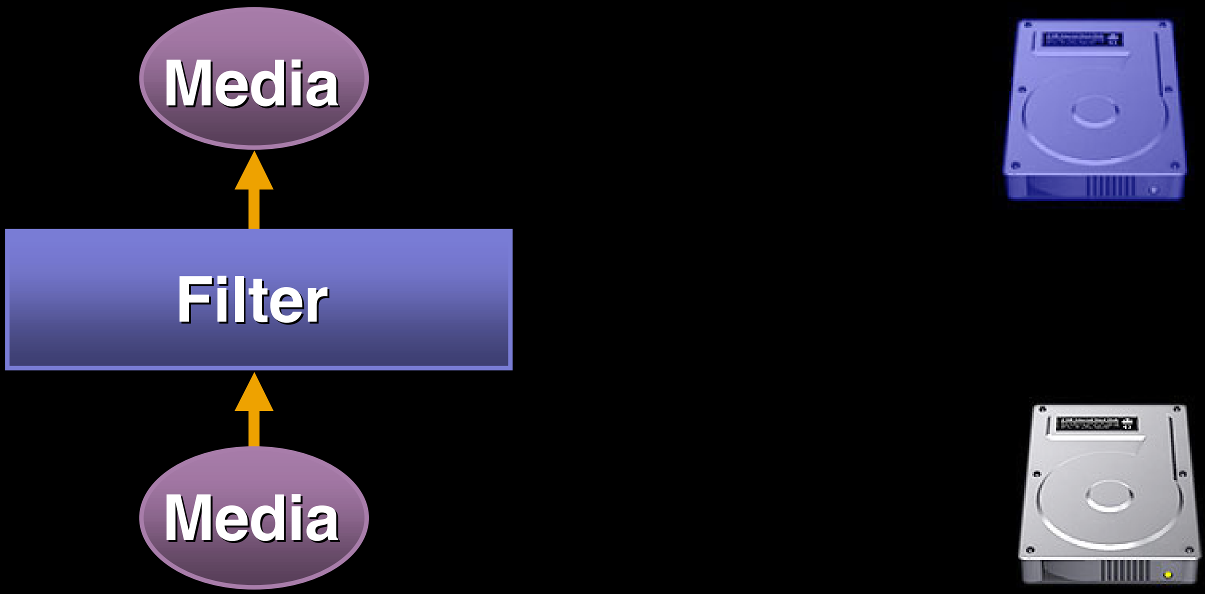
Example ioctl

```
char *                  bsdPath   = "/dev/rdisk0s9";
u_int32_t               bs = 0;
int                     fd;

if ( ( fd = open ( bsdPath, O_RDONLY, 0 ) ) != -1 )
{
   if ( ioctl ( fd, DKIOCGETBLOCKSIZE, &bs ) != -1 )
           printf ( "blockSize = %d\n", bs );

   close ( fd );
}
```

Media Filters

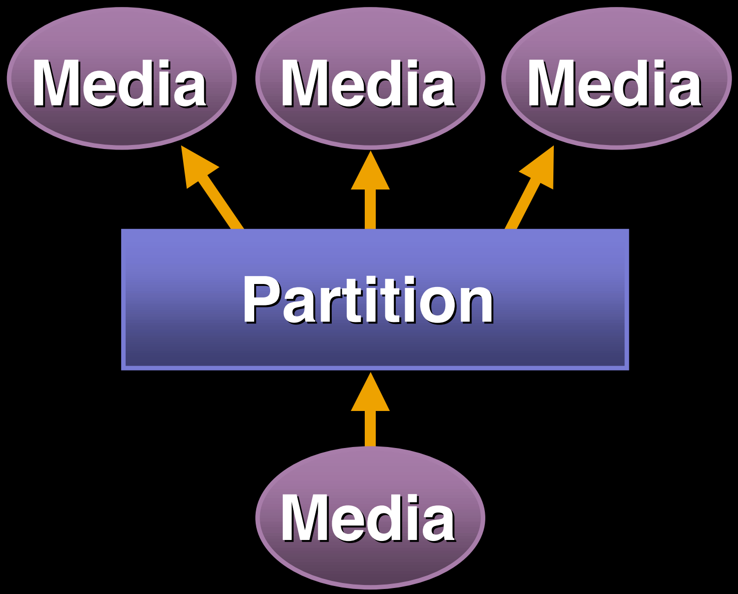
- Block oriented parsing (e.g., compression, encryption, etc.)
- Abstracted from hardware details
- Byte-based API with 64-bit parameters
  - See the *Inside Mac OS X: Writing Drivers for Mass Storage Devices*

Simple Media Filter



Media Partitioning Schemes

- Are nothing more than filters themselves



Media

Media

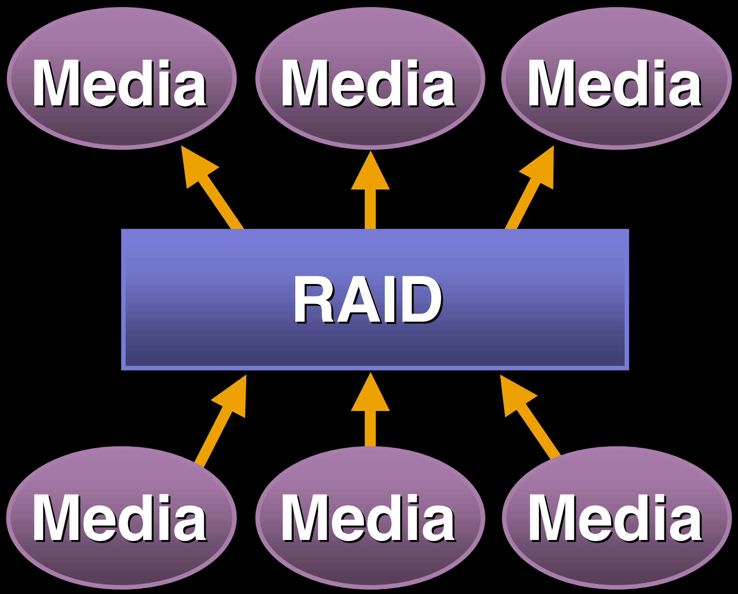
Media

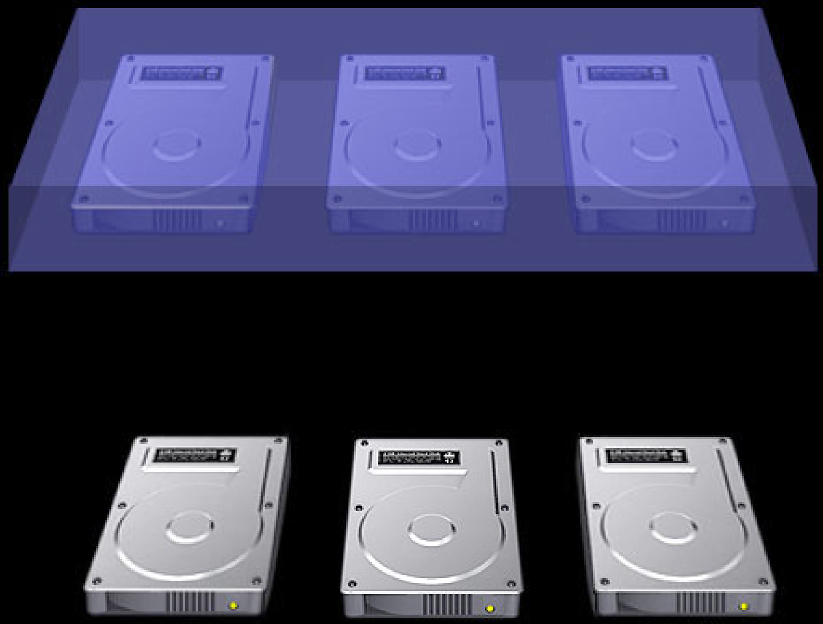
Partition

Media

Complex Filters

- Arbitrary Complexity
- Software RAID





Demo

Dan Preston
Mass Storage Demo Boy

Methods of Access: User Clients

Chris Sarcone
Software Engineer
Mass Storage

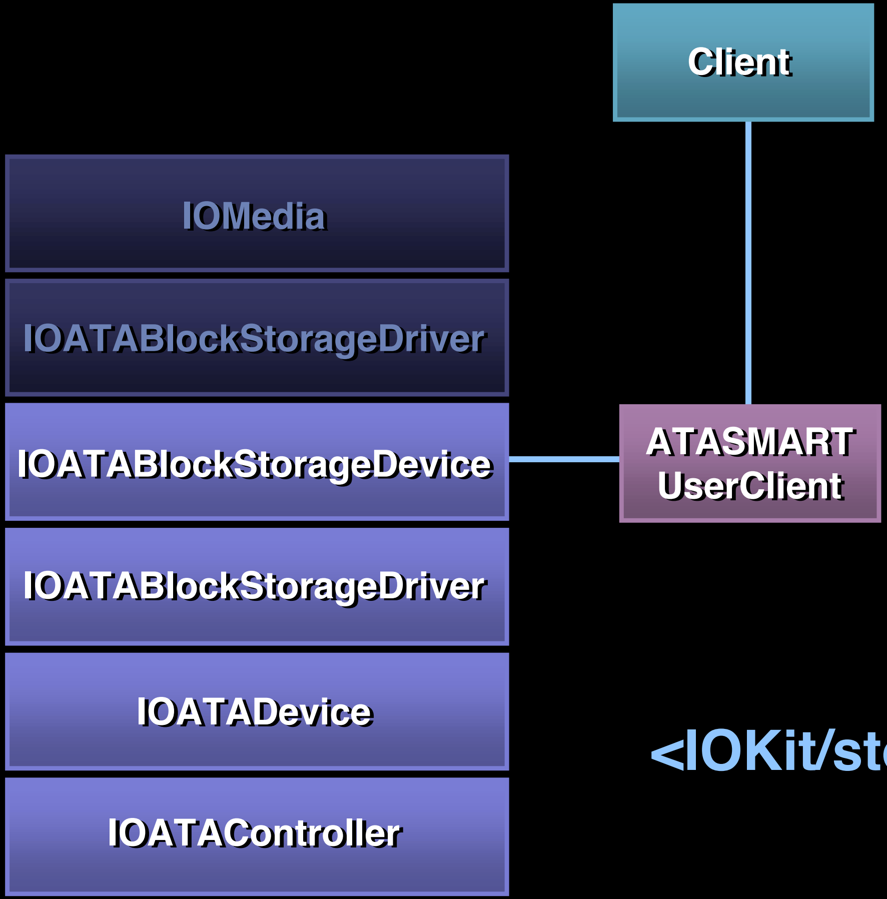
What Is a User Client

- An intermediary between a kernel object and a user space client
- Exports control to user space code via:
  - Device Interface
    - IOCFPlugIn defines the interface
    - Act as proxies for kernel objects
  - IOCTL
  - Other methods

Apple-Supplied User Clients

- Media layer
  - IOMediaBSDClient
  - IOCDMediaBSDClient
  - IODVDMediaBSDClient
- Device layer
  - ATASMARTUserClient
  - SCSITaskUserClient

User Client for ATA Devices



- Only for S.M.A.R.T.
- Device Interface
    - ATASMARTInterface
- Documentation forthcoming—For now, consult header doc in

<IOKit/storage/ata/ATASMARTLib.h>

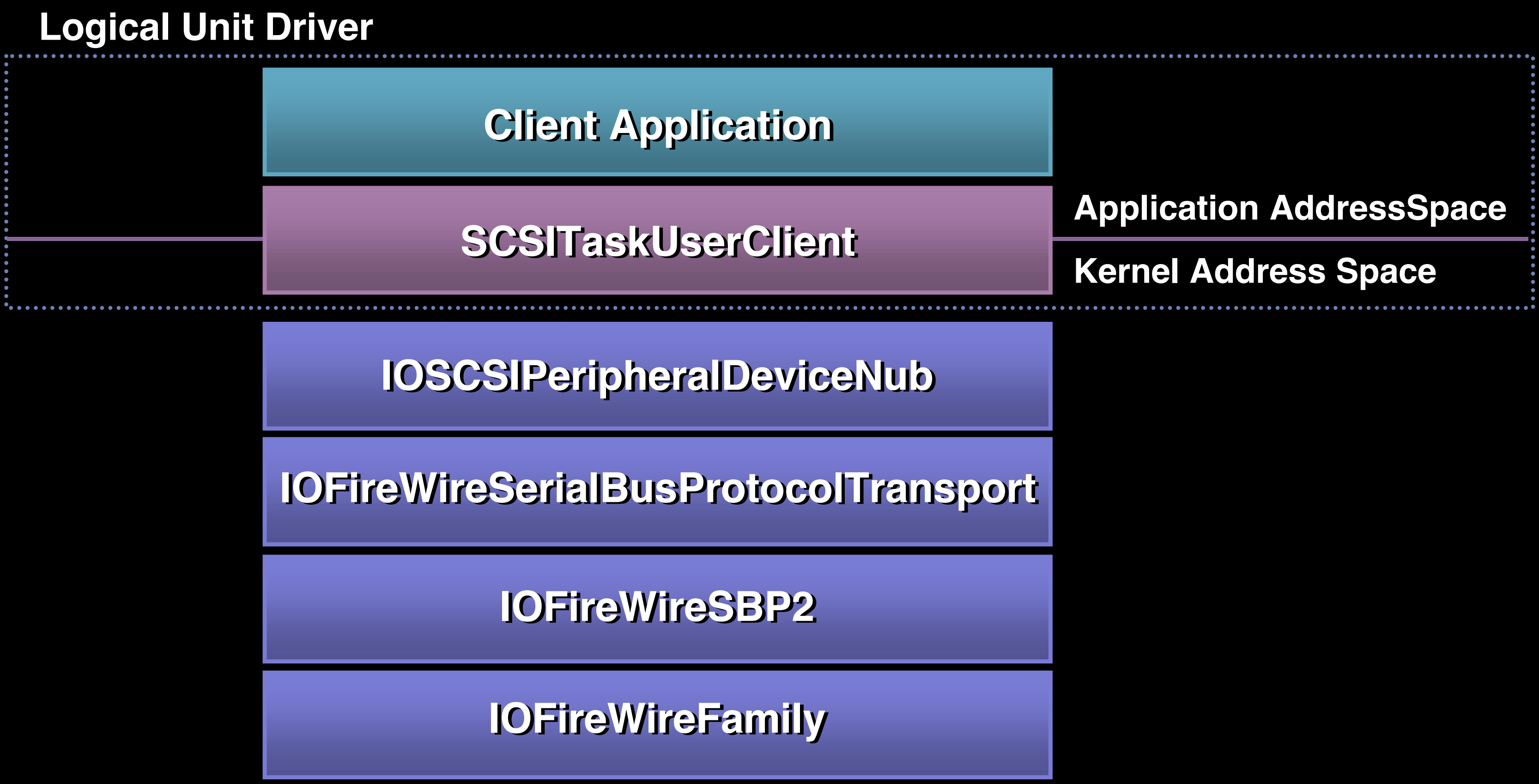
User Client for SCSI Devices

- Device Interfaces
  - SCSITaskDeviceInterface
  - MMCDeviceInterface
- Supplemental Interfaces
  - SCSITaskInterface
- Documented in *Inside Mac OS X: Accessing Hardware From Applications*
- SDK available
  (http://developer.apple.com/hardware)

User Client for SCSI Devices

- SCSITaskDeviceInterface
  - Peripheral device types not supported by Apple with in-kernel drivers
    - Tape drives
    - Scanners
    - Printers
  - Exclusive access model with full control of the device
  - SCSI Task Management Functions NEW

User Client for SCSI Devices



Logical Unit Driver

Client Application

SCSITaskUserClient

Application AddressSpace

Kernel Address Space

IOSCSIPeripheralDeviceNub

IOFireWireSerialBusProtocolTransport

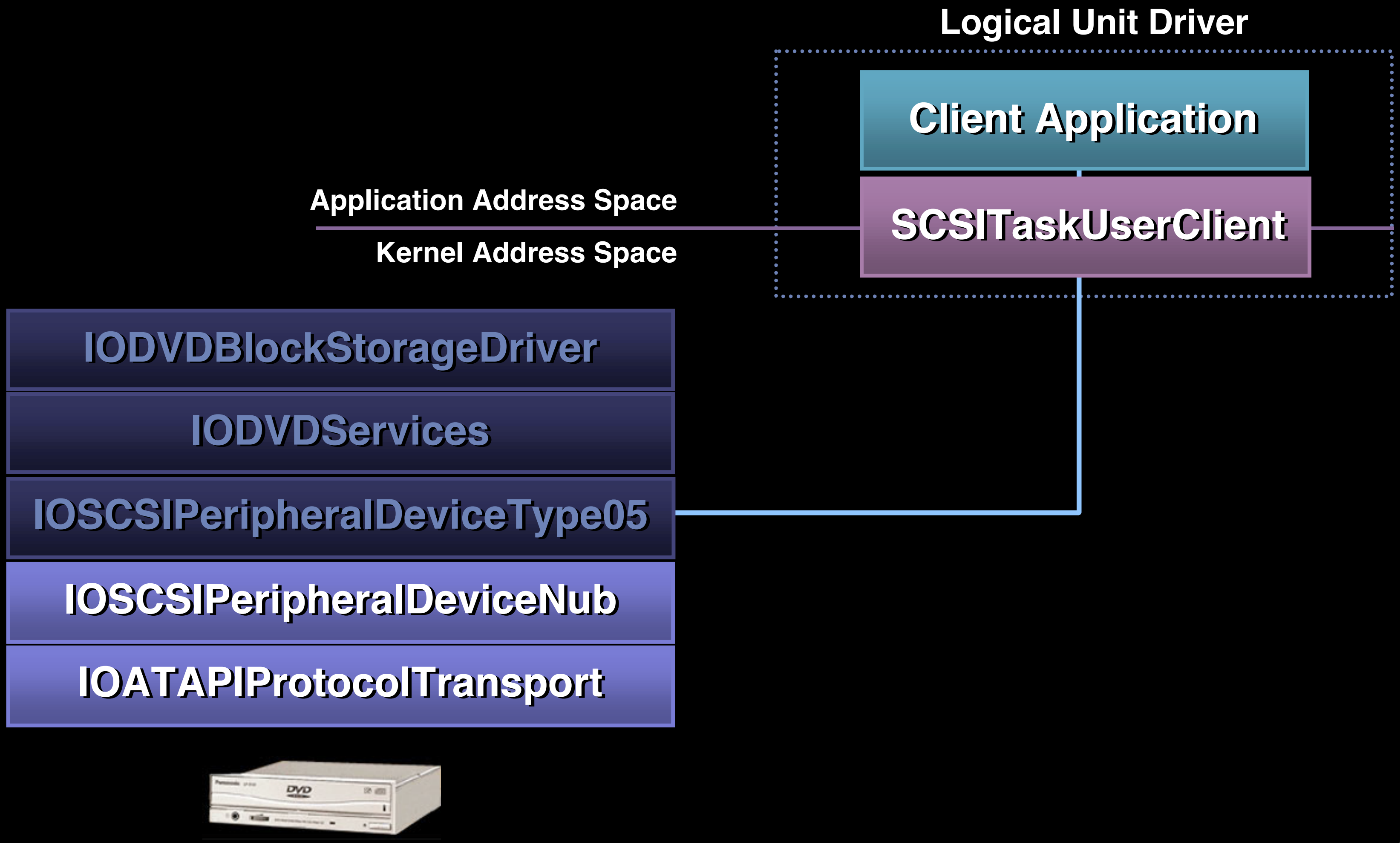
IOFireWireSBP2

IOFireWireFamily

User Client for SCSI Devices

- MMCDeviceInterface
    - Provided for all MMC-2 compliant drives capable of authoring
        - GetConfiguration profiles
        - Mechanical Capabilities mode page
    - Can be used in conjunction with SCSITaskDeviceInterface to gain exclusive access to authoring devices
    - Migrating some existing functionality to IOCTLs where appropriate

User Client for SCSI Devices



Obtaining Exclusive Access to a SCSI Device

- Use new Carbon APIs in Files.h to unmount volume
    - FSCreate/DisposeVolumeOperation
    - FSUnmountVolumeSync/Async
- Use Digital Hub for notifications of blank media insertion (‘dhub’ AppleEvent)

Obtaining Exclusive Access to a SCSI Device

- Use the SCSITaskDeviceInterface to query whether exclusive access is available
- Obtain exclusive access
- Once you have obtained exclusive access, your application is the Logical Unit Driver

Demo

Dan Preston
Mass Storage Demo Boy

Custom User Clients

- Is a user client really necessary?
  - How much data? How often?
  - Are there other ways which satisfy my needs?
- Consider other options
  - IORegistry properties
    - Finding static properties
    - Dynamically setting properties
  - Asynchronous notifications from the driver to interested parties

Custom User Clients

- Create a subclass of IOUserClient
- Define your API
- Implement the functions in the kernel
- Test with a command line tool or app

http://developer.apple.com/samplecode/Sample_Code/Devices_and_Hardware/IOKit/SimpleUserClient.htm

Methods of Access: IORegistry

Using the IORegistry Functions

```
// Get services which match our class
IOServiceGetMatchingServices (
        masterPort,
        IOServiceMatching ( “MyClassName” ),
        &iterator );
// Loop using iterator to get service objects
{
   // Call IORegistryEntryCreateCFProperties()
   // Inspect properties
   // Release properties dictionary
   // Release object ( iterator bumped its refcount)
}
```

Using the IORegistry Functions

```
// Get services which match our class
IOServiceGetMatchingServices (
            masterPort,
            IOServiceMatching ( “MyClassName” ),
            &iterator );
// Loop using iterator to get service objects
{
    // Fill in dictionary using CoreFoundation routines
    // Call IORegistryEntrySetCFProperties()
    // Release object since iterator bumped its refcount
}
```

Overriding setProperties()

```
IOReturn
MyClassName::setProperties ( OSObject * properties )
{
    OSDictionary *        dict = NULL;

    // Check if the property is of correct type
    dict = OSDynamicCast ( OSDictionary, properties );
    if ( dict != NULL )
    {
            // Check for properties the driver understands
    }
    return kIOReturnSuccess;
}
```

Methods of Access: Migration From Obsoleted Methods

Reasons to Migrate

- SCSIAction
  - Only IOKit should contain hardware APIs
    - Carbon will formally deprecate this API in Jaguar
    - Carbon will remove this API post-Jaguar
  - Performance is not great
- IOSCSILib/IOCDBLib
  - Will only work with IOSCSIFamily-based drivers
  - SCSITaskUserClient has equivalent or better functionality

Alternatives for Those Using Obsoleted Methods

- Image Capture Architecture-based drivers should be written for scanners in order to promote system-wide use for applications
- CUPS architecture-based drivers should be written for printers in order to promote system-wide use for applications

Roadmap

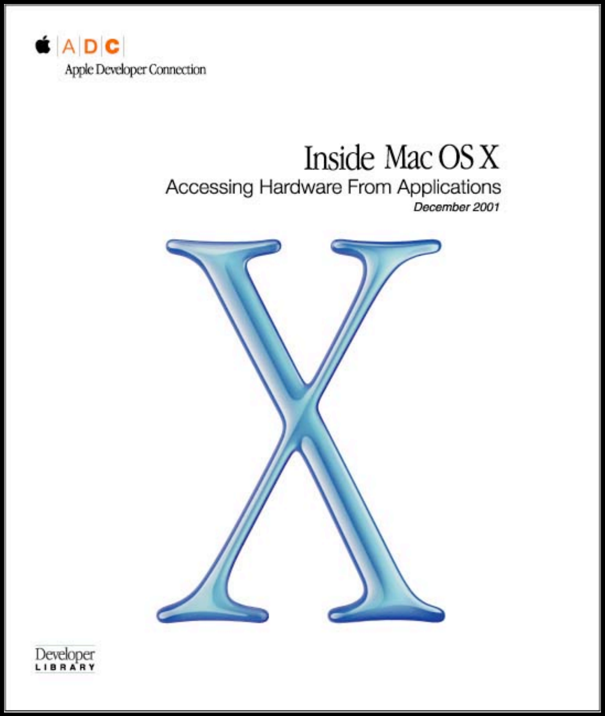
008 DiscRecording APIs: Write CDs and DVDs from your applications	Hall 2 **Thurs., 2:00pm**
510 Printing and Mac OS X: Access printers from your applications	Hall 2 **Thurs., 10:30am**
515 Image Capture Framework: Imaging device support for applications	Room C **Fri., 2:00pm**

For More Information

- SCSI Specifications **http://www.t10.org**
- ATA Specifications **http://www.t13.org**
- Mass Storage Discussion List
  - To subscribe send an e-mail to **requests@sam.apple.com**
  - With message subscribe x_mass_storage

Documentation

Accessing SCSI and ATA Devices



ADC
Apple Developer Connection

Inside Mac OS X
Accessing Hardware From Applications
December 2001

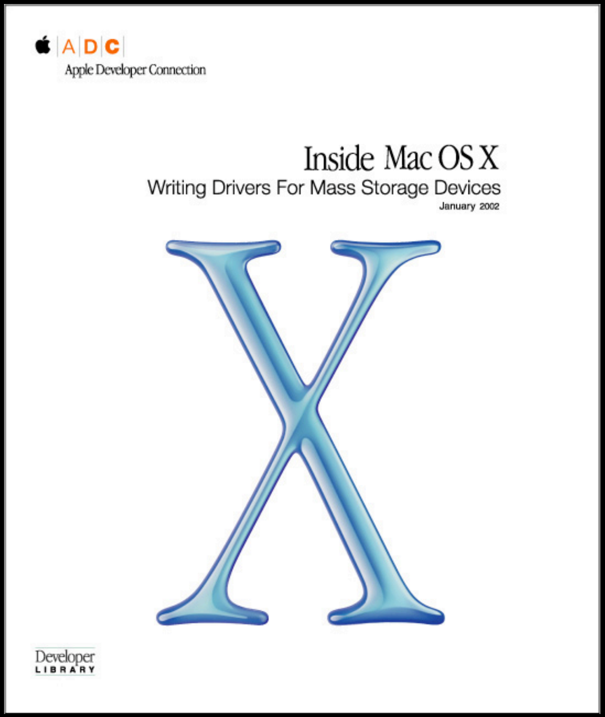
Developer
LIBRARY

- Accessing Hardware From Applications
    - Working With SCSI Architecture Model Devices

Documentation > Darwin > I/O Kit Documentation
Softcopy: http://developer.apple.com/techpubs/macosx/Darwin/IOKit/iokit.html
Hardcopy: http://www.vervante.com/apple

Documentation

Accessing SCSI and ATA Devices



ADC
Apple Developer Connection

Inside Mac OS X
Writing Drivers For Mass Storage Devices
January 2002

Developer
LIBRARY

- Writing Drivers for Mass Storage Devices

Documentation > Darwin > I/O Kit Documentation
Softcopy: http://developer.apple.com/techpubs/macosx/Darwin/IOKit/iokit.html
Hardcopy: http://www.vervante.com/apple

Who to Contact

Mark Tozer
Hardware Evangelist
tozer@apple.com

Q&A

```
/desktop: No such file or
[duarte:~] ryan% ls
Desktop    Library    Mus
Documents  Movies     Pic
[duarte:~] ryan% cd /Libr
[duarte:/Library] ryan% c
[duarte:/Library/Webserve
CGI-Executables  Document
```

Mark Tozer
Hardware Evangelist
tozer@apple.com

http://developer.apple.com/wwdc2002/urls.html

WWDC2002

WWDC2002

WWDC2002